Data encoding for SDL in ITU-T Rec. Z.104

Rick Reed

TSE Ltd

Purpose

Define data encoding for

- Communication between components
- Implementation by different tools
  - Versions
  - Platform variations
- Calculation/control of signal sizes

Z.104 features (1)

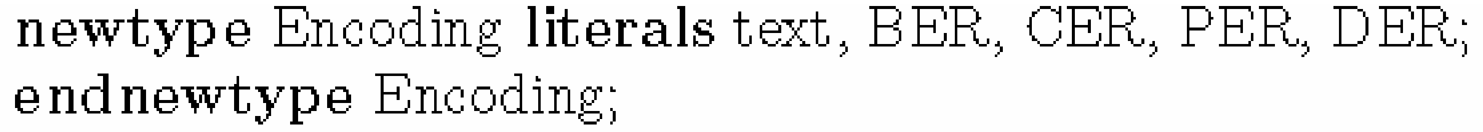
- Encode SDL data to a text string
- Encode ASN.1 based data
  - According to X.690 series
  - Must be
    - CHOICE
    - Signal names = choice names
  - As BIT STRING, OCTET STRING
- Object (reference) data not supported

Z.104 features (2)

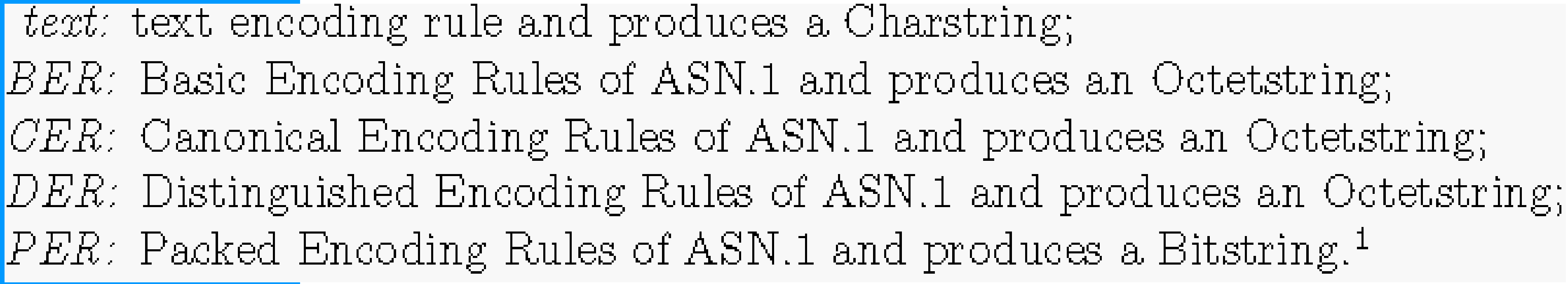
- Encoding on communication paths
- Implicit interfaces + signals from ASN.1
- Input, analyse, store encoded messages
- Output signals from encoded message
- Encode signal to encoded message
- Decode signal from encoded message
- Implied data definitions
- ASN.1 implied interface definition

Encoding rules

- **package** Predefined



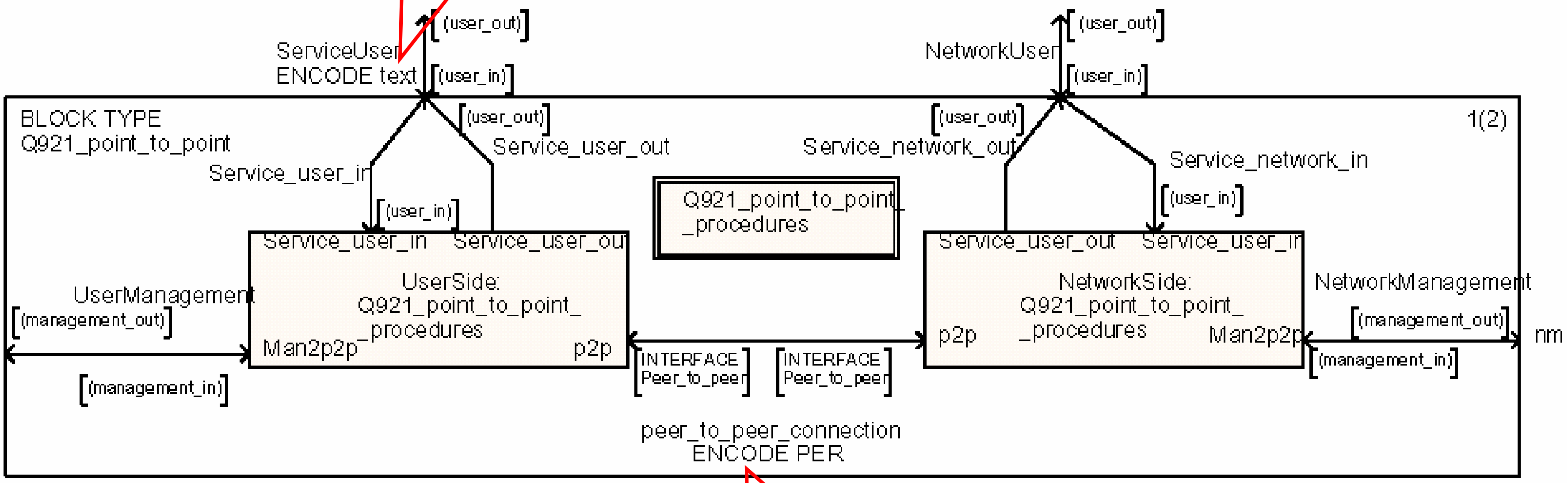
```
newtype Encoding literals text, BER, CER, PER, DER;
endnewtype Encoding;
```



text: text encoding rule and produces a Charstring;
BER: Basic Encoding Rules of ASN.1 and produces an Octetstring;
CER: Canonical Encoding Rules of ASN.1 and produces an Octetstring;
DER: Distinguished Encoding Rules of ASN.1 and produces an Octetstring;
PER: Packed Encoding Rules of ASN.1 and produces a Bitstring.[1]

- [1] There are 4 variations of `PER`
  - Aligned and Unaligned to Octet
  - ‘Basic’ and Canonical
- The choice of PER name(s) is tbd.

The Q.921 Example



Encoding on gate

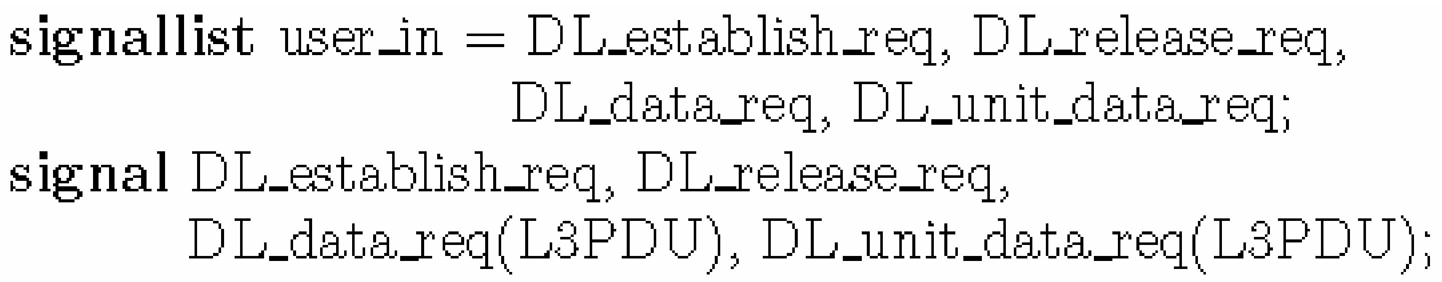
Encoding on channel

Encoding on

- gate ServiceUser
- channel peer_to_peer_connection

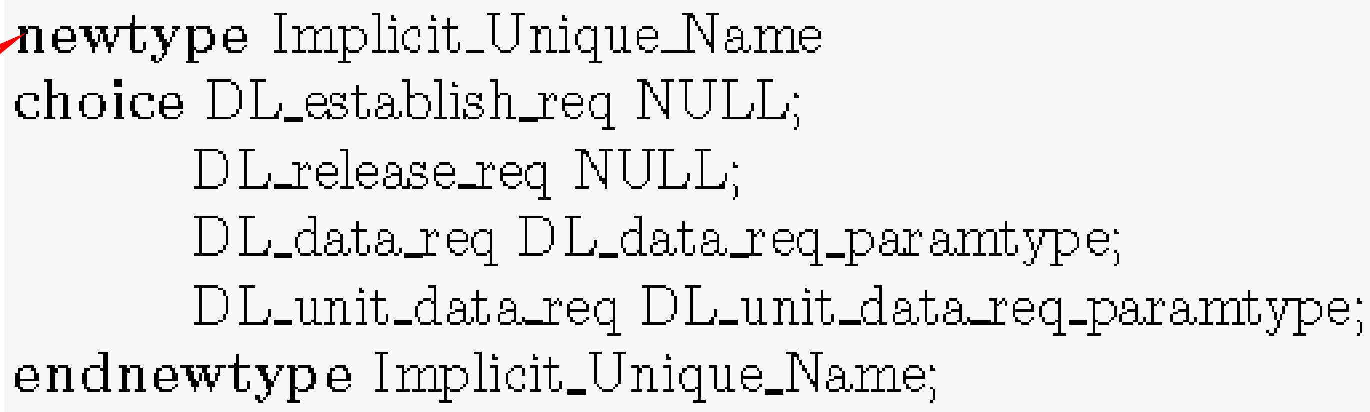
Path with encode:Implied data types

Implied by encode



```
signallist user_in = DL_establish_req, DL_release_req,
                     DL_data_req, DL_unit_data_req;
signal DL_establish_req, DL_release_req,
       DL_data_req(L3PDU), DL_unit_data_req(L3PDU);
```

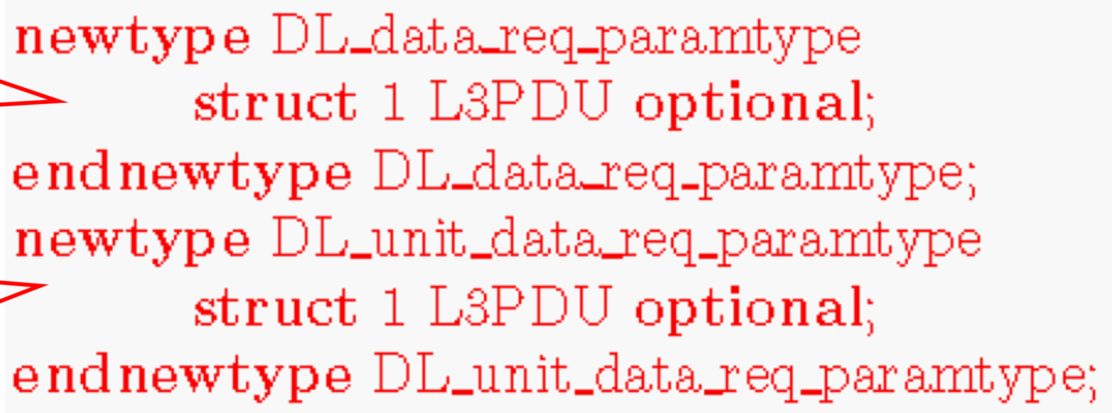
Encoding datatype



```
newtype Implicit_Unique_Name
choice DL_establish_req NULL;
       DL_release_req NULL;
       DL_data_req DL_data_req_paramtype;
       DL_unit_data_req DL_unit_data_req_paramtype;
endnewtype Implicit_Unique_Name;
```

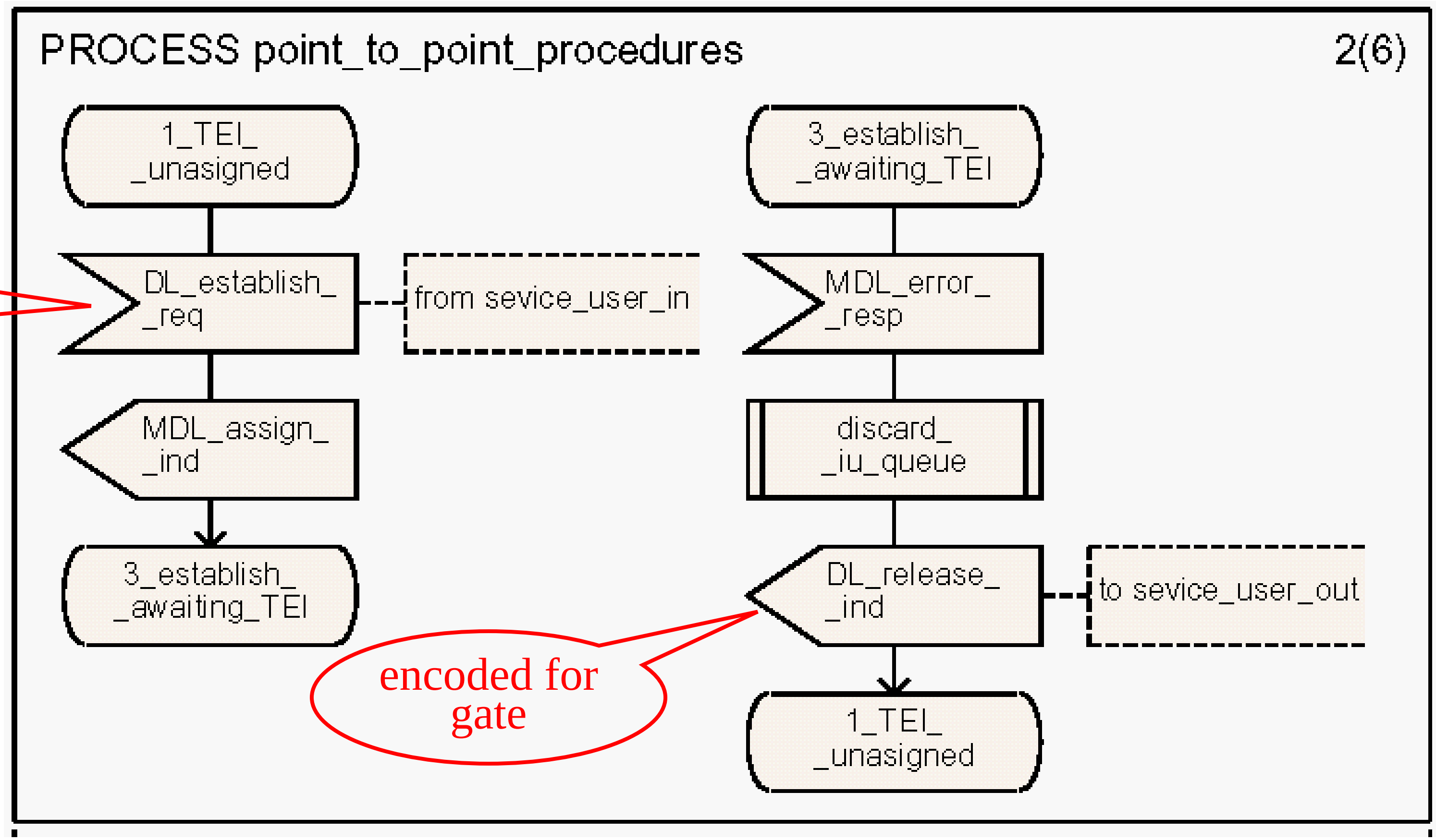
Implied datatype for DL_data_req parameters

Implied datatype for DL_unit_data_ _req parameters



```
newtype DL_data_req_paramtype
        struct 1 L3PDU optional;
endnewtype DL_data_req_paramtype;
newtype DL_unit_data_req_paramtype
        struct 1 L3PDU optional;
endnewtype DL_unit_data_req_paramtype;
```

Encoded path Input + Output



decoded from gate

encoded for gate

- No special syntax for input/output
- Encoding only in Userside context

Text encoding

Basic types

- Boolean, Character, Charstring, Integer, Real, Duration, Time, Bit, Bitstring, Octetstring, NULL

Composite types

- String, Array, Vector, Powerset, Bag, STRUCT, CHOICE, *enumerated types*

Processing identity types

- Pid, *pid types*
  - ApplicationDefined, Integer, Octetstring, Bitstring, Charstring and a composite

    {**struct** identity Charstring; instance Natural}
- Object - not supported

Encoding a signal as text

```
signal DL_establish_req, DL_release_req,
       DL_data_req(L3PDU), DL_unit_data_req(L3PDU);
```

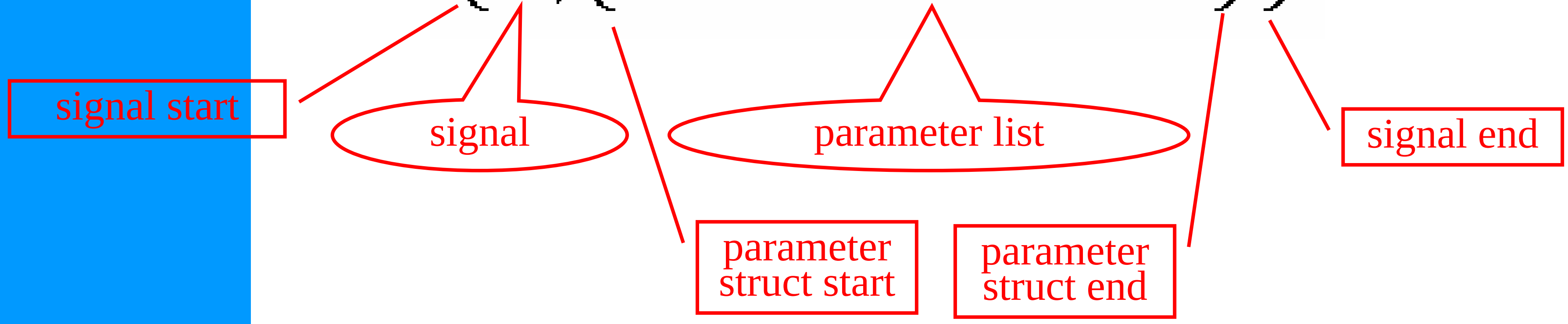
```
syntype L3PDU = Octetstring endsyntype L3PDU;
```

- Signal

$$DL\_data\_req('12ADCDEF'H)$$

- Encoding

$$\{2,\{'12ADCDEF'\}\}$$



signal start

signal

parameter list

signal end

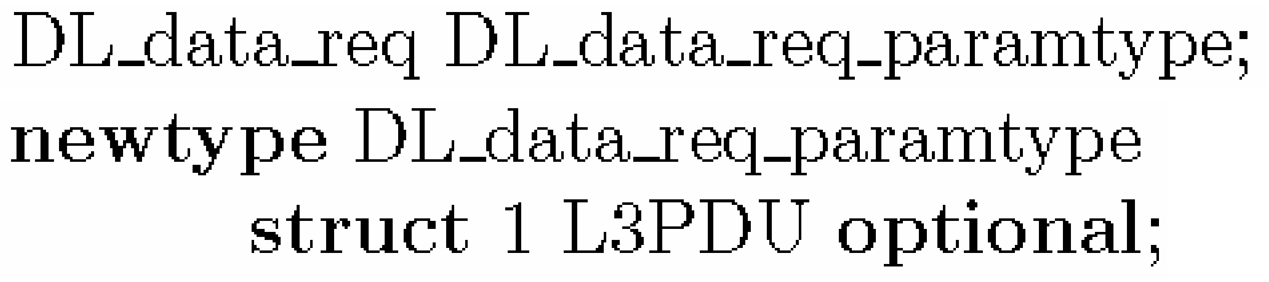
parameter struct start

parameter struct end

Commas separate parameters

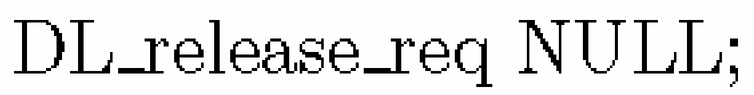
Text is not intended to be human readable

Missing Parameters and Null



```
DL_data_req DL_data_req_paramtype;
newtype DL_data_req_paramtype
        struct 1 L3PDU optional;
```

- DL_data_req with no parameters
  - {2,{}}

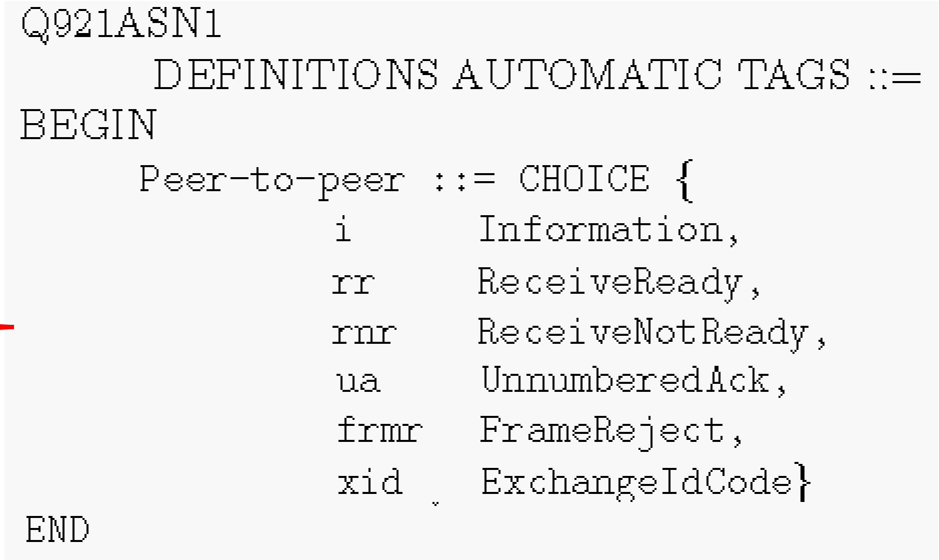


```
DL_release_req NULL;
```

- Data type NULL(such as DL_release_req)
  - Full encoding {1,0}
  - Omit NULL parameter {1,}
  - No parameter {1}

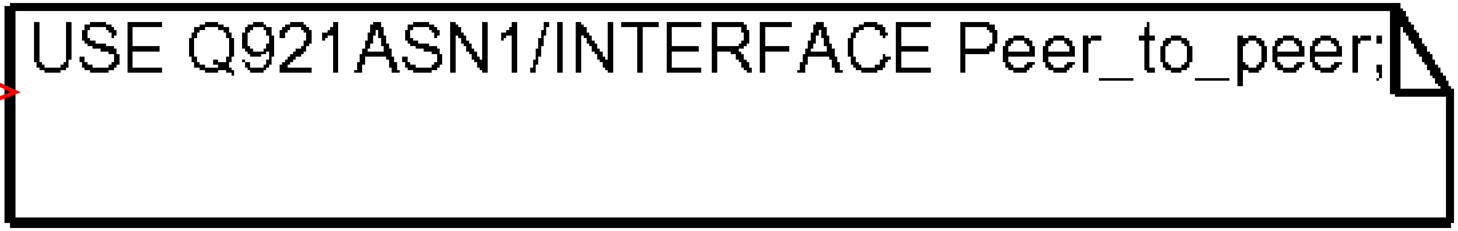
USE ASN.1+Implicit Interface

ASN.1 definition



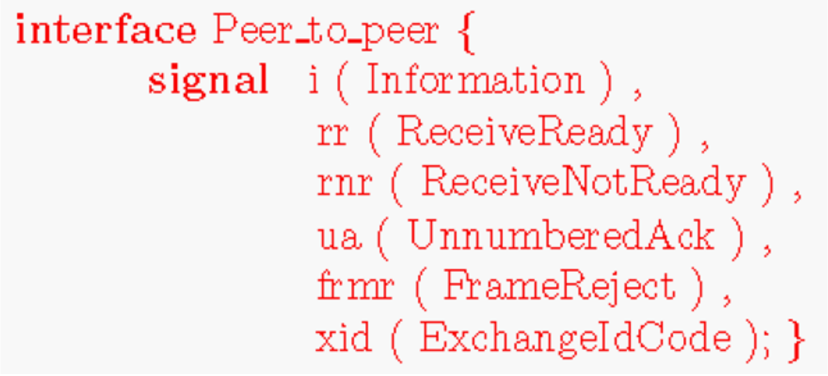
```
Q921ASN1
        DEFINITIONS AUTOMATIC TAGS ::=
BEGIN
        Peer-to-peer  ::= CHOICE {
                    i         Information,
                    rr        ReceiveReady,
                    rnr       ReceiveNotReady,
                    ua        UnnumberedAck,
                    frmr      FrameReject,
                    xid       ExchangeIdCode}
END
```

package use Clause for ASN.1



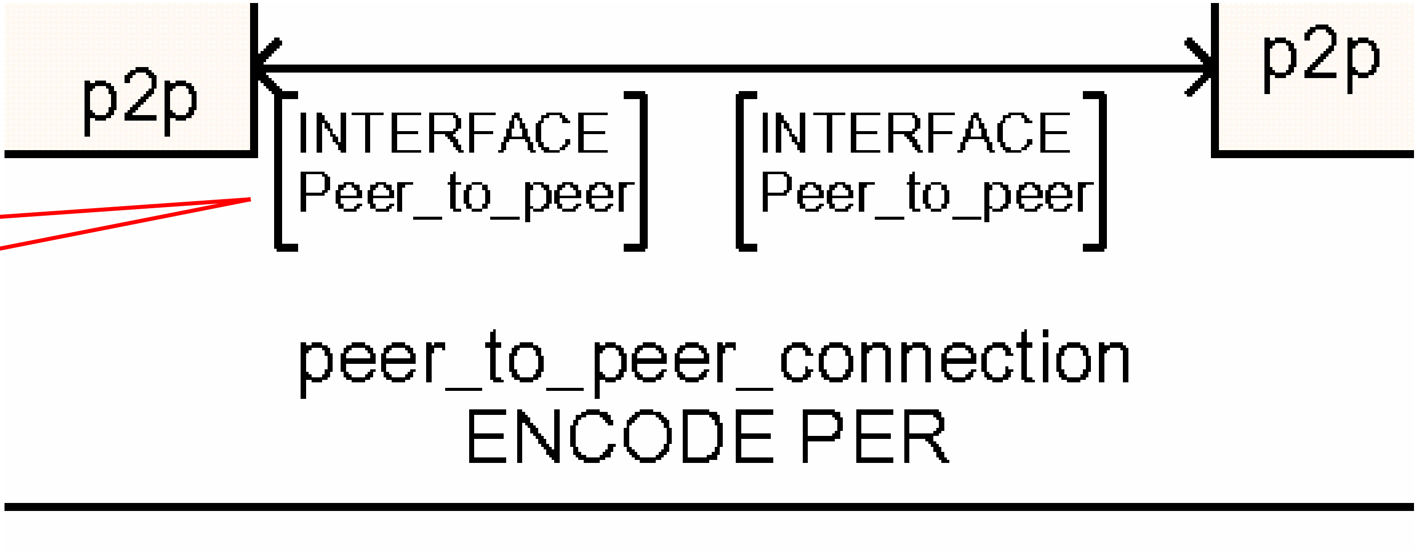
USE Q921ASN1/INTERFACE Peer_to_peer;

Implied interface and signals



```
interface Peer_to_peer {
        signal  i ( Information ) ,
                rr ( ReceiveReady ) ,
                rnr ( ReceiveNotReady ) ,
                ua ( UnnumberedAck ) ,
                frmr ( FrameReject ) ,
                xid ( ExchangeIdCode ); }
```

Using ASN.1 on a path



p2p

p2p

INTERFACE
Peer_to_peer

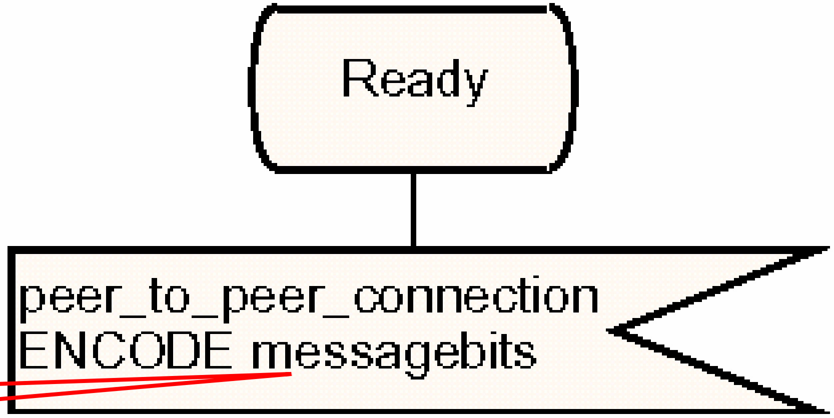
INTERFACE
Peer_to_peer

peer_to_peer_connection
ENCODE PER

Using the
implied
interface

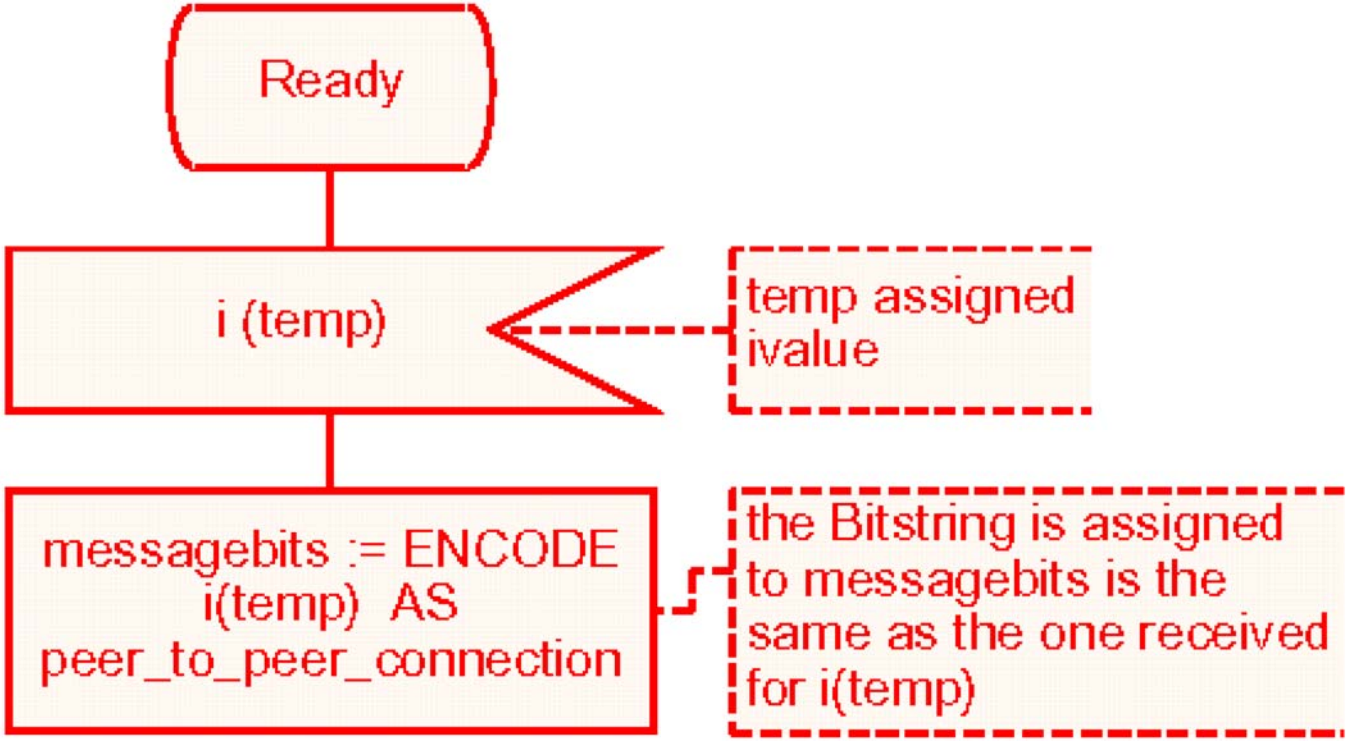
- USE implies Peer_to_peer interface
- Interface with encoding implies data types for encoding

Input without decode



Communication path

string variable encoded data


Ready
i (temp)
temp assigned ivalue
messagebits := ENCODE i(temp) AS peer_to_peer_connection
the Bitstring is assigned to messagebits is the same as the one received for i(temp)

Explanation in terms of receiving a value then encoding a signal

implicit decode

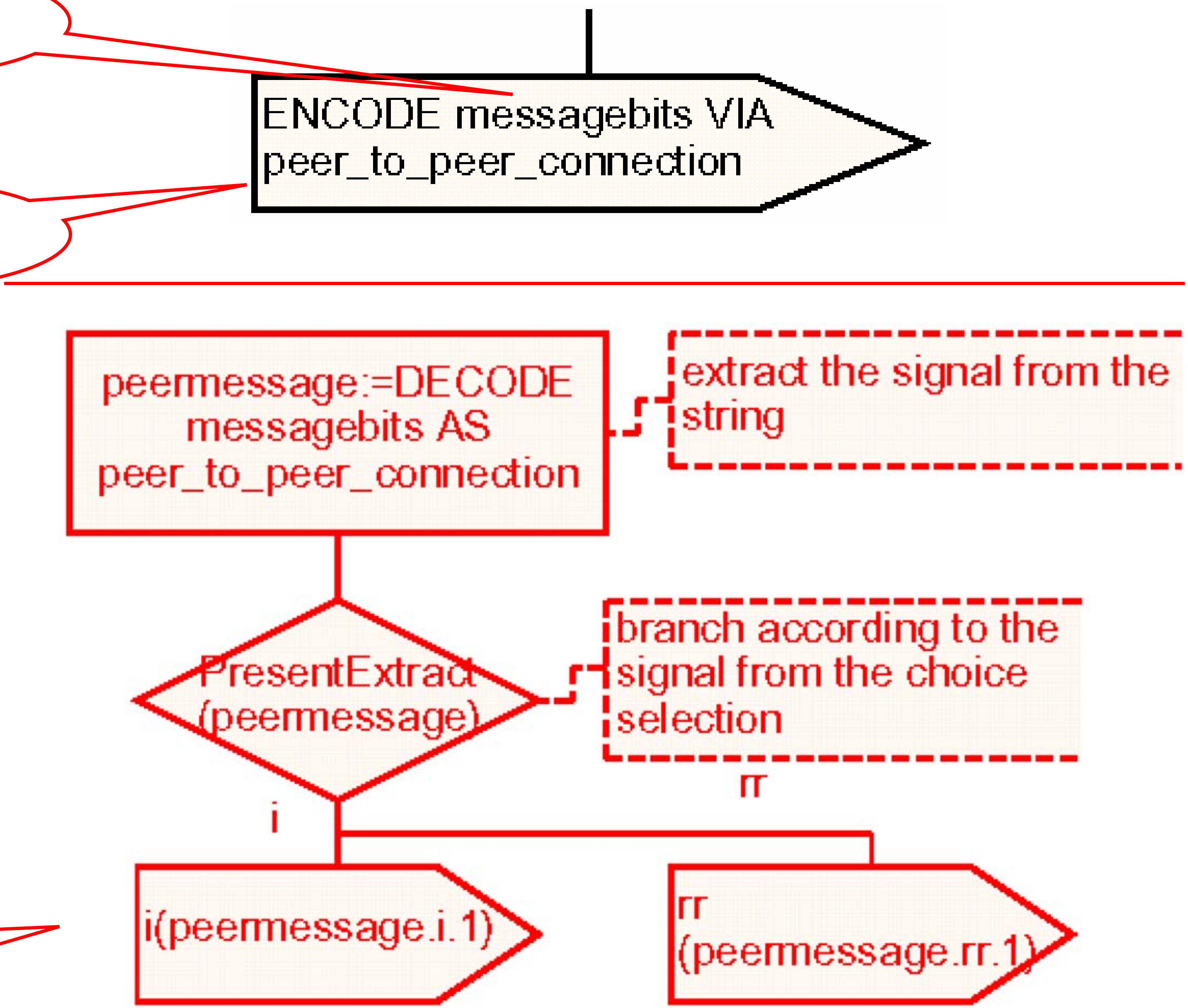
Output of encoded message

string for encoded data

Communication path

Explanation in terms of decoding the signal from stored data, followed by output

implicit encode



ENCODE messagebits VIA peer_to_peer_connection

peermessage:=DECODE messagebits AS peer_to_peer_connection

extract the signal from the string

PresentExtract (peermessage)

branch according to the signal from the choice selection

i

rr

i(peermessage.i.1)

rr (peermessage.rr.1)

ENCODE / DECODE expressions

```
decode matching_string_expression as path_id;

encode signal_id( parameters ) as path_id;
```

- DECODE stored string from input
- ENCODE to string for output
- Can be used for encapsulation
- May require dummy gates

Conclusion

- Some limitations - possible extensions
  - Other encoding rules (such as XER)
  - Application defined encoding rules
  - Input ENCODE when no rule on path
  - Output/encode from string expression
  - Output/encode from CHOICE expression
  - Other data formalisms than ASN.1
  - X.690 support for SDL
- Feedback requested (by 15th July 2004)
- Consent for approval 19-21July 2004